MDBS III®

ABSOLUTE POWER

USER'S MANUAL

mdbs®
THE KNOWLEDGE MANAGEMENT COMPANY

MICRO DATA BASE SYSTEMS, INC., P.O. BOX 248, LAFAYETTE, INDIANA 47902

MDBS III
SYSTEM SPECIFIC MANUAL
for
UNIX™ System V with C
(SPERRY 5000 SERIES)

Micro Data Base Systems, Inc.

P. O. Box 248

Lafayette, Indiana 47902

USA

PRELIMINARY

Version 3.09
September, 1987

Copyright Notice

This entire manual is provided for the use of the customer and the customer's employees. The entire contents have been copyrighted by Micro Data Base Systems, Inc., and reproduction by any means is prohibited except as permitted in a written agreement with Micro Data Base Systems, Inc.

Trademarks: MDBS is a registered trademark of Micro Data Base Systems, Inc.
UNIX is a trademark of AT & T Bell Laoratories.

© 1987 Micro Data Base Systems, Inc.

PREFACE

This manual describes the operating procedures for MDBS III extended network data base software within the UNIX SYSTEM V operating system. This manual concentrates on elements unique to the interaction among MDBS III, UNIX SYSTEM V, and the programming languages used with MDBS III.

The document consists of four parts:

- File Name Conventions - describing file names from the **mdbs** perspective
- Installing MDBS III Software - transferring MDBS III software to your VMS environment
- Running MDBS III Software - Starting MDBS software packages on your system
- MDBS III Data Base Administration - Implementation Strategy behind the multi-user version of MDBS III.

You should be familiar with the MDBS III product components. The manual assumes you know the data manipulation language (DML) commands and other MDBS III components and presents only their syntactic forms; command descriptions and component functions are described in detail in the appropriate MDBS III reference manuals.

TABLE OF CONTENTS

I. MDBS III Files I-1
File Names for MDBS Software I-1
File Name Conventions I-3
Default File Names for MDBS III I-3

II. Installing MDBS III Software II-1

III. Running MDBS III Software III-1
Special Key Definitions III-1
Command Arguments III-2
Data Description Language Analyzer (DDL) III-3
Data Base Control System III-4
Additional Packages III-6
Interactive Data Manipulation Language (IDML) III-6
Query Retrieval System (QRS) III-7
Report Definition Language (RDL) III-7
Design Modification Utility (DMU) III-8
Utilities III-8

IV. Multi-user Administration IV-1
Multi-user Data Base Control System IV-2
Apllication Process IV-2
Multi-User DML Commands IV-2
Recovery Procedures IV-3
Using the Data BAse Monitor (DBMTR) IV-3

Appendix

C Language A-1

PRELIMINARY

© Copyright 19876 Micro Data Base Systems, Inc.

FILE NAMES

MDBS III data base multi-user software exists for several operating environments; one of these supported environments is the Sperry 5000 series computer systems running the UNIX operating system.

The MDBS III software consists of several files on your distribution medium. To acquaint you with the software, this section describes the file names of your MDBS III software, the file name conventions, and the definition of a fully qualified file name.

FILE NAMES FOR MDBS SOFTWARE

Most files distributed with your MDBS III system are independent of the programming language; they relate to the various software modules of the MDBS III product family. The following list identifies the files associated with each MDBS III module.

DDL Files

ddl	Data Description Language (DDL) Analyzer module
sample.ddl	Sample DDL source text (schema) file

DMU Files

dmu	MDBS Design Modification Utility (DMU) module

PRELIMINARY

DMS Files

mdbsdms	Data Management System (DMS) - standard form
mdbsrtl	Data Management System (DMS) - transaction logging form
rcv	Transaction Recovery Utility

IDML Files

idml	Interactive Data Manipulation Language (IDML) module
util.err	Error message file for MDBS III QRS and IDML

QRS Files

qrs	Query Retrieval System (QRS) module
util.err	Error message file for MDBS III QRS and IDML

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

RDL Files

rdl	Report Definition Language (RDL) module
rdlsm	RDL sort/merge utility
librdl.a	RDL library
sample.rdl	Sample RDL specification

Utility Files

dmser	DMS error message utility
dms.err	DMS command status error message file
dbmtr	Data Base Monitor utility
install	MDBS III installation script

DBRS Files

dbrs	MDBS III Data Base Restructuring System (DBRS)
decomp	MDBS III DBRS data base decomposition utility
regen	MDBS III DBRS data base regeneration utility

Miscellaneous Files

read.me	If present, this text file contains last minute information **Please read this file**
serial.no	serial number of this MDBS III system
level.no	version number and creation date of this MDBS III system
chksums	checksums of MDBS III system distribution files

Language Specific Files

Certain files supplied with your MDBS III system depend on the programming language you use in your application. For each language interface you purchase, **mdbs** provides up to four language specific files:

sample.???	Sample source program using MDBS III DML commands
test.dms???	DMS test command file
libdmsm???.a	DML command library (standard form)
librtlm???.a	DML command library (transaction logging form)

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

In the language dependent files, the wild-card characters (???) are coded according to the MDBS III language interface; the other characters in the file names are valid. Replace the wild-card characters with the letter C for the C language interface.

Note: The **librtlm???.a** file is provided only if you purchased the recovery and transaction logging option of MDBS III.

FILE NAME CONVENTIONS

A fully qualified file name in this environment is compatible with UNIX SYSTEM V release 1R1. It consists of a valid UNIX SYSTEM V file name optionally prefaced by a directory path. The maximum length of the fully-qualified file name (file name + path name) is 128 characters.

For example, if **trial.ddl** is an UNIX SYSTEM V file in the directory (path) **/usr/ben**, the fully-qualified file name is:

/usr/ben/trial.ddl

If the path name is omitted from a fully-qualified file name, then the current (default) directory is assumed.

DEFAULT FILE NAMES FOR MDBS III

If a file name is not specified for the main data base area in the identification section of a DDL specification, the main area is assigned to a file having the same name as the data base. If a file name is not specified for an area defined in the area section of a DDL specification, then that area is assigned to a file having the same name as the area. No file name extensions are incorporated into these default file names. The default file names use upper case letters.

If a file name is specified without an extension, MDBS III uses one of the default extensions:

Extension	Designation
.DB	main data base area file
.DBA	additional data base area file
.DDL	text file written by MDBS.DDL
.LOG	transaction log file
.TXT	file written by MDBS.QRS WRITE or SPEW command
.$$$	MDBS.RCV scratch files

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

SYSTEM INSTALLATION AND TESTING

MDBS III data base software runs in several operating environments, including the Sperry 5000 Series computers using the UNIX operating system. The MDBS III software modules are distributed as several files on floppy diskettes or as a magnetic tape (9 track, 1600 bytes per inch, and blocking factor of 20) written with the UNIX file archive utility (**cpio**). The installation procedure uses a shell script called **install**.

The **mdbs** installation copies MDBS III software files from the distribution media to the hard disk of your system with the UNIX **cpio** command.

Follow these steps to install MDBS III:

1. Log in as the root user.

2. Use the **cd** command to change the default directory to the directory where you want to initially place MDBS files. This should be an empty directory or the temporary file directory (**/tmp**). For example, to initially place files in the temporary directory, enter:

```
cd /tmp
```

3. Now use the **umask** command to change the default protection mask so that MDBS files will be copied with the proper permissions:

```
umask 0
```

PRELIMINARY

Note: This **umask** value remains in effect until you log out or enter another umask command.

4. Now copy the software from the distribution media to your hard disk.

- For a distribution tape, mount the tape onto the tape drive and enter the following command to copy the software from the distribution tape to your hard disk:

```
cpio -iBcmdvu </dev/rtp
```

- For distribution diskettes, insert the first diskette into the floppy disk dirve and enter the following command to copy the software from the diskette to your hard disk:

```
cpio -iBcmdvu </dev/rfd70
```

Repeat this procedure for each diskette in the set, then continue with the next step.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

5. If it is present, you should read the **read.me** file included with the software so as to be aware of any recent changes. Use the UNIX **cat** command to show it on the screen:

cat read.me

Note: If you suspect an MDBS file has been corrupted or copied improperly, verify its checksum with the **chksums** file and the UNIX **sum** utility.

6. You can now install MDBS for system-wide usage or for use in the local directory only.

- For system-wide use, **install** copies the MDBS files into the appropriate system directories where all users have access to them. By default the **install** command places MDBS executables in the directory **/usr/bin**, MDBS support files in **/usr/lib**, and MDBS include files (if any) in **/usr/include**. To run **install**, enter the command:

install

- For a local installation, **install** copies the MDBS files into the target directory. Specifically, **install** places executables in **directory/bin**, support files in **directory/lib**, and include files in **directory/include**. To run **install**, enter the the command:

install directory

TESTING MDBS SOFTWARE

After installation, the sample DDL specification and sample application program can be used to test the success of the installation procedure. This is accomplished by entering:

test.dmsc

The test program executes MDBS.DDL to generate the data dictionary and initialize the data base for the sample DDL specification; the sample application program's source code is then compiled and linked with MDBS.DMS. The MDBS Data Management System data base control system and interprocess communication manager (**mdbsdms**) is then executed; after this the compiled sample program is executed and its menu appears, allowing the user to load data into the data base and to retrieve data from it. If this is successful then MDBS.DDL and MDBS.DMS have been properly installed and may now be used as described in Chapters III and VI.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Note: Since **install** creates new copies of the executable, library, and include files, two copies of these files are present on your system.The copies in the present directory (where MDBS was first installed) can be deleted after testing is complete.

Sample files are also left in the present directory. You can delete them or move them to another directory if you wish to save them.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

RUNNING MDBS III SOFTWARE

To run MDBS III software, type the software module name at the prompt of the UNIX shell and press the return key. This starts the appropriate software package with its default option settings. There are a few other helpful hints in running MDBS III software modules:

- special function keys for interactive MDBS III modules
- command arguments

The interactive MDBS III software modules designate assign functions to specific keys. The next section itemizes each "function key" and its use in these software modules.

Each MDBS III software module has command arguments that let you specify optional settings. Some arguments are common to all the MDBS software modules; others exist only for one or two modules. The command arguments for each module are presented below.

SPECIAL KEYS FOR INTERACTIVE MDBS SOFTWARE

Interactive MDBS III software modules recognize certain keystrokes to have special significance. Default keystrokes with corresponding definitions are listed below. They are implementation-dependent and may vary from installation to installation.

return terminates an input line; also called ***enter*** or ***newline***, equivalent to ***ctrl–m***

kill stops a line entry and restarts the input line; default **@**, often set to ***ctrl–u***

erase deletes the last character typed; default **#**, often set to backspace (***ctrl–h***)

tab causes a tab character to be placed in the line; equivalent to ***ctrl–i***

quit returns control to the UNIX command interpreter (hard interrupt); default ***ctrl–***, sometimes ***ctrl–|***

intr causes the prompt of the interactive software to appear (soft interrupt); default ***del*** (***ctrl–?***), sometimes ***ctrl–c***

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

COMMAND ARGUMENTS

The command arguments allow you to identify items such as the data base user, the workspace size, and the data base you want to access when running a software module. To specify a command argument, include a dash (-) and the command argument letter on the command line. Separate the argument from the command name and other command arguments with one or more spaces. When additional information is specified with the option such as a data base name or a number, that information immediately follows the command argument letter.

Most MDBS products use the following common command arguments:

-u*username* specifies a user name.
-p*password* specifies a password.
-d*dbname* specifies a data base name.
-b*nnnn* specifies the number of bytes allocated for working space.
-m mutes the MDBS banner display and other non-error messages.

For example, to invoke MDBS-IDML using the user name **user**, the password **pass**, on the data base named **JOBS** and allocating 32000 bytes of working space, this command might be used:

idml -uuser -ppass -dJOBS -b32000

Any of these options can be omitted; if the -u, -p, or -d options are omitted, the program prompts you for the user name, password, or data base name. The password will be not be shown on the screen if you are prompted for it.

If the -b option is omitted the program will use a large default value for the working space size.

An octal, decimal, or hexadecimal base may be used when specifying numbers with command-line options such as -B.

An octal number begins with the digit 0 and has no other digits greater than 7. Example: octal 0777 is decimal 511. Any number beginning with 0 and does not contain the digits 8 or 9 is assumed to be an octal number.

A decimal number begins with one of the digits 1-9 and/or contains the digits 8 or 9.

A hexadecimal number has at least one of these characteristics:

- begins with 0x (Example: hex 0x12 is decimal 18)
- ends with H or h (Example: hex 400H is decimal 1024)
- contains the digits A-F or a-f (Example: hex 0D is decimal 13)

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

DATA DESCRIPTION LANGUAGE (DDL)

The UNIX command for running MDBS.DDL is:

ddl ***filename*** **—b*nnnn***

filename and **–b*nnnn*** are optional arguments.

If ***filename*** is omitted, then the MDBS.DDL program responds with the **::** prompt and is ready for interactive usage (see VI-A,B of the MDBS III Data Base Design Reference Manual).

If a file name is specified, MDBS.DDL runs in batch mode (see VI-C of the MDBS III Data Base Design Reference Manual). The ***filename*** must be a fully-qualified-file-name. The contents of ***filename*** must be a valid DDL specification. For example, use the following command to analyze the DDL source text contained in the file **trial.ddl** in the directory **/usr/ben**:

ddl /usr/ben/trial.ddl

Note: Enclose any file name specified in a DDL source text in double quotes (" ").

The optional **—b** argument is a byte allocation argument that determines the size in bytes (***nnnn***) of DDL's working area. **—b** is normally not needed for DDL in this environment and can be ignored.

When MDBS.DDL is invoked interactively, it is possible to execute operating system commands without exiting the interactive DDL command mode. When the double colon (**::**) prompt appears, the user can execute an operating system command by pressing the **!** key and then entering the operating system command. This is a convenient method to use when creating DDL descriptions, since you can use the system editor to edit DDL descriptions. For example, the following shows how a user might invoke the screen editor **vi** from within DDL:

```
::!vi file.ddl
        •
        •
        •
editing the DDL specification on file.ddl
        •
        •
        •
        :x
::init file.ddl
```

Note: The **vi** screen editor may not be available on all systems.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Pages per area default

If the number of pages for an area is not stated in a DDL specification, then the default value of 50 pages is used. This default is for both the main area and any extra areas.

Page size

If the page size for an area is not stated in a DDL specification, then the default value of 512 bytes per page is used. This default is for both the main area and any extra areas.

The page size for any area must be at least 256 bytes.

Maximum Number of Areas and Maximum Pages per Area

The maximum number of MDBS III areas per database under UNIX is 8. The maximum number of pages per area is 8191 pages in each of 8 areas.

DATA BASE CONTROL SYSTEM

All MDBS III data base systems in the UNIX environment are multi-user systems that use a data base manager process. The data base manager must be active before any application program can access the data base. The data base control system has two forms:

Standard Form	- supports all DML commands in a multi-user environment (mdbsdms)
Transaction Logging Form	- adds transaction logging functions to the supported DML commands (mdbsrtl)

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Standard Form

Once you have installed MDBS III, start the data base control system (MDMS) with two steps:

1. Log in as the user who you wish to have ultimate control over the data base. Normally this is the user who owns the data base file(s).

2. Enter the command:

mdbsdms *filename*

The ***filename*** argument is required. ***Filename*** is the name of the physical file containing the main area of this MDBS data base, as specified in the DDL source specification for the data base. For example, to start the data management system for a data base contained in the file **/usr/data/customers.db**, enter:

mdbsdms /usr/data/customers.db

Normally, the data base management system uses the entire remaining space of its memory segment for program buffers. If you want the data base management system to use less space, specify the amount you want it to use with the -b option previously described in this section.

Transaction Logging Form

Start the data base control system with transaction logging (MRTL) with two steps:

1. Log in as the user who you wish to have ultimate control over the data base. Normally this is the user who owns the data base file(s).

2. Enter the command:

mdbsrtl **—l***logname filename*

The -l flag is optional. If -l is used, ***logname*** is the name of the log file to which all data base transactions are logged. If -l is omitted, the log file name declared in the data base's DDL source specification is used.

The ***filename*** argument is required. ***Filename*** is the name of the physical file containing the main area of this MDBS data base, as specified in the DDL source specification for the data base.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Normally, the data base management system uses the entire remaining space of its memory segment for program buffers. If you want the data base management system to use less space, specify the amount you want it to use with the -b option previously described in this section .

ADDITIONAL PACKAGES

MDBS III add on packages use the same start up conventions described earlier. Information specific to each package is given below.

Interactive Data Manipulation Language (IDML)

To run the IDML program, enter this command from the operating system:

```
idml
```

You can optionally specify the name of an alternative startup file and/or common options listed at the beginning of this chapter. The default startup file must have the name STARTUP. If an alternative file name is used on the command line, it must be fully qualified as stated in section I.

For example, to use the startup information on the file idml.go and allocate 8000 bytes, use the operating system command line:

```
idml idml.go -b8000
```

Operating system commands may be used within IDML by entering the command preceded by an exclamation mark (!).

To temporarily leave IDML, enter an exclamation mark and press return. This initiates a new copy of your command language interpreter as a subprocess of IDML. When you logout of this subprocess, you will return to IDML.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Query Retrieval System (QRS)

To invoke the interactive MDBS-QRS program, use the following operating system command line:

qrs

The user can optionally specify the name of an alternative startup file and/or common options listed at the beginning of this chapter. The default startup file must have the name STARTUP. If an alternative file name is used on the command line, it must be fully qualified (see section I).

For example, to use the startup information on the file qrs.go and allocate 8000 bytes, enter the command line:

qrs qrs.go –b8000

Notes: A fully qualified file name must be used with the QRS READ command and with the FN parameter. When the READ command is used, the file name must be in quotes (either single or double).

Operating system commands may be used within QRS by entering the command preceded by an exclamation mark (!).

To temporarily leave QRS, enter an exclamation mark and press return. This initiates a new copy of your command language interpreter as a subprocess of QRS. When you logout of this subprocess, you will return to QRS.

Report Definition Language (RDL)

Note: RDL requires a C language interface to the MDBS III DMS.

To invoke RDL, use the following operating system command line:

rdl *filename*

The ***filename*** is the name of a file containing a valid RDL source specification. The -u, -p, -d, and -b options listed under Command Arguments may be used with RDL.

To compile the C program(s) generated by an RDL run, enter:

crpg

To run the programs generated by an RDL run, enter:

rpg

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Design Modification Utility (DMU)

To invoke the interactive MDBS.DMU program, use the following operating system command line:

```
dmu
```

MDBS-DMU is a single user program. It should not be used on a database that other users may be accessing.

UTILITIES

This section describes the start up procedures and command arguments of each utility that accompanies your MDBS III system:

- DMS Command Status (DMSER) - lists meanings DMS command statuses
- Recovery Utility (RCV) - processes transaction log for data bases

DMS Command Status (DMSER)

The DMSER program is a utility that lists the text for the DMS command status numbers.

To obtain the text for command status 31, for instance, enter:

```
dmser 31

DMS command status      31
Insufficient room in memory
```

To obtain the text for all command status numbers, enter:

```
dmser
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Recovery Utility (RCV)

The RCV utility is described in the MDBS-RTL manual. A log file used with the RTL form of MDBS must be a fully qualified file name within UNIX. The log buffer size in this environment is 512 bytes. To invoke the interactive MDBS.RCV program, use the following operating system command line is used:

```
rcv
```

Be sure to make all necessary backups before using RCV. The **-m** option mutes the MDBS banner that appears at invocation. The **-i** option can be used to specify an input file.

RCV is a single-user program. It should not be used on a database that is accessible to other users.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MULTI-USER ADMINISTRATION

Unrelated tasks using an MDBS III data base in the UNIX environment communicate with the data base by using shared memory (see the entries shmctl, shmget, and shmop in section 2 - System Calls - of the Sperry 5000 and 7000 Series Programmer Reference Manual, Volume 1). The shared memory is created by the data base control system (mdbsdms or mdbsrtl) and is shared by each application process. Application processes directly interact with this shareable memory region to retrieve and update the contents of a data base. A diagram of the multi-user data base system is shown below:

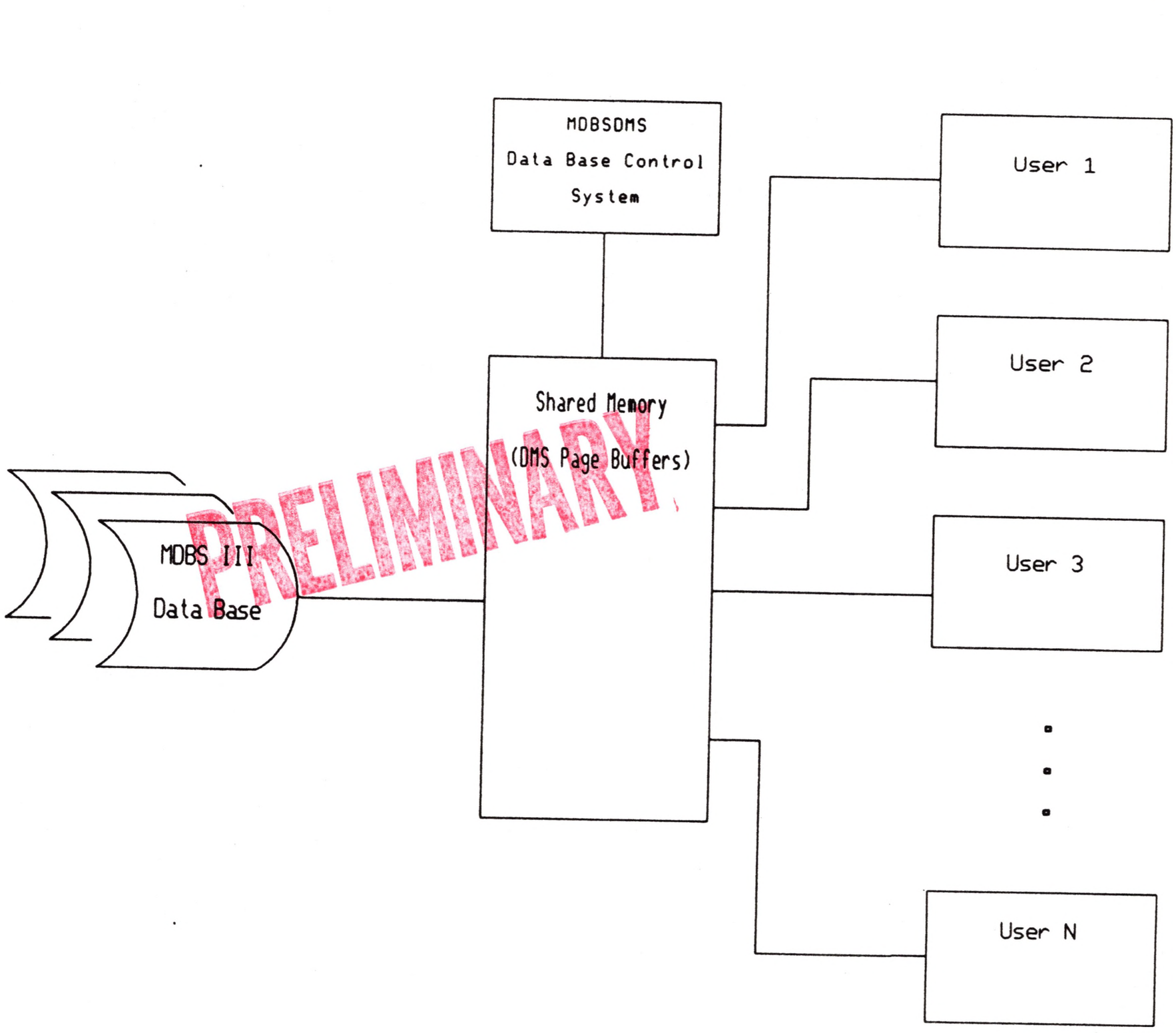

Figure X-1: Multi-user Access to an MDBS III Data Base

The MDBS III software co-ordinates the memory usage among several application processes and removes the need to tailor application programs to the details of the operating environment.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

MULTI-USER DATA BASE CONTROL SYSTEM

The multi-user data base control system (mdbsdms) manages the environment for one data base. mdbsdms manages the data base environment in the following ways:

- Initializes the shared memory used by application programs. This section of memory contains things such as the data base page buffers and the currency indicators of each application using the data base
- Opens the data base and transaction log file
- Periodically brings the data base to consistency by flushing page buffers
- Periodically verifies the active processes that are using the data base. The data base control system removes any extraneous references from nonexistent processes
- Closes the data base and performs other housekeeping tasks

APPLICATION PROCESS

Application processes use the data base contents. Each application process interacts directly with a portion of the shared memory to retrieve information from the data base and to store information within the data base. Changes to the data base are written directly to data base page buffers and to log file buffers (when transaction logging is in use).

The data base control system allows spawned processes to access and manipulate the data base; however, the data base control system can have trouble determining when the child process ends (trapping **sigchld**). If your application forks off a child process, make the parent process sleep until the child process terminates. Observing this caution will help maintain the integrity of your data base.

MULTI-USER DML COMMANDS

The multi-user commands described in Chapter XV of the MDBS III DMS Reference Manual are available in a multi-user environment.

The time unit used with UNIX for the DMS Contention Count command (MCC) is 1 second.

The Data Base Statistics command (DBSTAT) displays statistics for the shared section of memory, not for an individual application process.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

RECOVERY PROCEDURES

Since page posting is generally not meaningful in a multi-user environment, posting is not allowed with the multi-user version of MDBS III. Thus the PIFD and TRABT commands should not be used. If used, they result in a command status message of 34 (no such routine), but have no other effects.

All run units log transactions to a single log file, which is specified by the multi-user data base control system. Placing the log file specification with mdbsrtl means the LGFILE DML command is not available in a multi-user environment. If used, the LGFILE command results in a command status of 34.

USING THE DATA BASE MONITOR (DBMTR)

DBMTR is an interactive multi-user administration utility that can be executed simultaneously with various run units. To start DBMTR, enter the following command on the operating system command line:

```
dbmtr

MDBS Multi-User Data System Monitor          Ver. 3.xx
(C) Copyright 19xx Micro Data Base Systems, Inc.

List of available data base files
.
.
.
```

If only one data base is available, it opens that one. Otherwise it prompts you for a data base file name and option selection:

```
Database file name:

Option Summary:
        Stats: display operational statistics
        Reset: reset communications
        Close: close data base and shut down communications
        Access: access a different data base
        TRSYNC: transaction log file synchronization
        Quit: exit monitor

Option (Stats, Reset, Close, Access, Trsync, Quit) ?
```

Each option is described below.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Stats

The Stats function retrieves and displays data base paging statistics for the global section of memory. These statistics are identical to those provided by the DBSTAT command (see the MDBS III DMS Reference Manual).

In addition, the Stats function lists the process identifier for each run unit that can access the data base.

Reset

The Reset option closes and restores the interprocess communication link with the global memory section. Use this function when a process inadvertently terminates or aborts.

Close

This function closes the data base and stops the mdbsdms process.

Access

The access function transfers communications from one open data base to another without leaving DBMTR.

Trsync

The TRSYNC function synchornizes the transaction log file by forcing all complex transactions to complete. This function is identical to the TRSYNC command described in the MDBS III RTL Reference Manual.

Quit

This function exits the multi-user administration monitor.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

Batch Usage of DBMTR

Any of the six options described above can be specified on the command line that invokes DBMTR, by indicating the first letter of the desired option. For instance,

dbmtr –d*filename* **–s**

executes DBMTR with the statistics option. When an option is specified on the command line, the menu is not displayed and there is an automatic exit from DBMTR after the option's task is completed.

Multiple options can be specified with no separators. The options are processed in the order in which they are specified. (If one of the options causes processing to terminate, then the remaining specified options are ignored.) For instance,

dbmtr –d*filename* **–ls**

performs the logout option and then the statistics option, before automatically exiting from DBMTR.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

APPENDIX A

C LANGUAGE

This appendix presents the C language specific issues for MDBS III with the UNIX operating system. The appendix uses the following structure:

MDBS III Language Dependent Files A-1

Application Programs A-1
- Compiling and Linking Application Programs A-1
- Symbol Conflicts in Application Programs A-2
- Running Application Programs A-2

Data Exchange A-3
- Non-numeric Data Items A-3
- Integer Data Items A-4
- Unsigned Data Items A-5
- Internal Decimal Data Items A-6
- Real Data Items A-7
- Repeating Data Items A-8

DML Command Forms A-9
- DML Command Status A-9
- Special Commands A-9
- Open and Close Commands A-11
- Find Commands A-11
- Retrieval and Modify Commands A-12
- Assignment Commands A-13
- Schema Access Commands A-14
- Utility Commands A-15
- Boolean Commands A-16

DML Command Syntax A-17

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C LANGUAGE APPENDIX

MDBS III LANGUAGE DEPENDENT FILES

MDBS III in the UNIX environment has three files (four files if you purchased RTL) unique to the C programming language interface:

sample.c	Sample C program using DMS functions
test.dmsc	MDBS-DMS test command file
libdmsmc.a	MDBS-DMS library (standard form)
librtlmc.a	MDBS-DMS library (RTL form)

APPLICATION PROGRAMS

Creating an executable application program consists of compiling the program successfully and linking the compiled module(s) with the appropriate MDBS III DMS library (libdmsmc.a or librtlmc.a). The linking process joins only those routines needed from the DMS library to the compiled module. This section describes the procedures to compile, link, and run application programs using MDBS III.

Compiling and Linking an Application Program

You can choose between two methods to compile and link application programs using MDBS III:

- use a local library of MDBS.DMS (in the current directory)
- use a standard system library of MDBS.DMS

The install program normally makes MDBS.DMS available as a standard library.

Local DMS Library

If MDBS.DMS is in the current directory, compile and link your program with the command:

cc *program.c* **libdmsmc.a** **—o** *program*

program.c is application program name.

libdmsmc.a is the multi-user MDBS.DMS library (in the current directory).

program is the executable application program.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Standard System DMS Library

If MDBS.DMS is put in a standard system library, then compile and link your program with following command:

cc *program.c* **–ldmsmc –o** *program*

program.c is application program name,

dmsmc is the multi-user MDBS.DMS library (in **/usr/lib**),

program is the executable application program.

Symbol Conflict in Application Programs

The symbols used by MDBS DML commands are translated to avoid conflict with variables or functions used by application programs. Variable or function names containing the uppercase letters "MDBS" should be avoided in user programs. In addition, some MDBS routines use system routines such as **malloc()** and **open()**.

Note: User programs should not use system routine names for function or variable names. This redefines a system routine and can cause unpredictable program behavior.

Running an Application Program

To run *program*, enter the following command at the operating system prompt:

program

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

DATA EXCHANGE

This section shows the type, size, and value correspondences that exist between MDBS data items and C variables. Correct usage of DML creation (**create**), modification (**put**), and retrieval (**get**) commands depends on a knowledge of these correspondences. Other DML commands, such as finding a record also require input from a C variable, where the variable must be consistent with a data item of a particular type and size.

Non-numeric Data Items

This table shows the correspondence between MDBS non-numeric data types and C language data types.

MDBS Data Item		C Variable	
type	size	type	size
binary	n	character	n
character	n	character	n
string	n	character *(with ASCII null byte terminator)*	n+1
date	—	character	10
time	—	character	9

Strings in C are terminated with a null byte (a character having the ASCII value 0). MDBS strips off the null byte when storing a string. The null byte is restored when an MDBS string is read back into a C string. **All C strings should be declared with a length one greater than the expected maximum length, in order to allow for the extra null byte.**

Strings are compressed when stored in an MDBS data base. Characters with ASCII decimal values less than 32 (the blank character) are converted to blanks.

Characters with values ranging from ASCII 32 to ASCII 127 are stored and sorted normally. Characters having an ASCII value greater than 127 are not converted; however, such characters may not sort in the expected order.

Tables showing conversion between MDBS and C numeric data types start on the next page.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Integer Data Items

MDBS III integer data items are consistent with C **int** variables. The interrelationships among MDBS III integer sizes and the host language variables are listed in Table 1. This table contains three main items:

- correlation of MDBS III integer data item sizes to C **int** variables
- data limits when storing information in the data base
- data limits when retrieving information from the dat base

As an example, when storing a data value from an integer C variable into a 1 byte integer data item, the value must be in the range -128 to 127. Any other value for the integer C variable will not be permitted and the DML command that attempts to store such a value will return a command status 33, "**Data conversion error**".

When retrieving a data value from a 3 byte integer data item into a long integer C variable, an appropriate value in the range -8388608 to 8388607 is deposited in the C variable.

As a third example, suppose we want to store the value 2147483645 into a 4 byte integer data item. This is accomplished with a **put** command that uses a long integer C variable having the value 2147483645.

Table 1. Integer Correspondences

MDBS Integer size	C Variable type	Storing Data: Source C variable range	Storing Data: Target data item range	Retrieving Data: Source data item range	Retrieving Data: Target C variable range
1	short int	-32768 to -129 -128 to 127 128 to 32767	undefined -128 to 127 undefined	-128 to 127	-128 to 127
2	short int	-32768 to 32767	-32768 to 32767	-32768 to 32767	-32768 to 32767
3	long int	-2147483648 to -8388609 -8388608 to 8388607 8388608 to 2147483647	undefined -8388608 to 8388607 undefined	-8388608 to 8388607	-8388608 to 8388607
4	long int	-2147483648 to 2147483647	-2147483648 to 2147483647	-2147483648 to 2147483647	-2147483648 to 2147483647
n>4	long int	-2147483648 to 2147483647	-2147483648 to 2147483647	< -2147483648 -2147483648 to 2147483647 > 2147483647	undefined -2147483648 to 2147483647 undefined

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Unsigned Data Items

MDBS III unsigned data items are consistent with C **unsigned** variables. The interrelationships among MDBS III unsigned sizes and the host language variables are listed in Table 2. This table contains three main items:

- correlation of MDBS III unsigned data item sizes to C **unsigned** variables
- data limits when storing information in the data base
- data limits when retrieving information from the dat base

As an example, when storing a data value from an unsigned C variable into a 1 byte unsigned data item, the value must be in the range 0 to 255. If the value is in the range 256 to 65535, it will not be permitted and the DML command that attempts to store such a value will return a command status of 33, **"Data Conversion Error"**.

When retrieving a data value from a 3 byte unsigned data item into a long C variable, an appropriate value in the range 0 to 8388607 is deposited into the C variable.

As a third example, suppose we want to store the value 2147483649 into a 4 byte unsigned data item. This is accomplished with a **put** command that uses a unsigned C variable having the value 2147483649.

Table 2. Unsigned Correspondences

MDBS Unsigned size	C Variable type	Storing Data		Retrieving Data	
		Source C variable range	Target data item range	Source data item range	Target C variable range
1	unsigned short	0 to 255 256 to 65535	0 to 255 undefined	0 to 255	0 to 255
2	unsigned short	0 to 65535	0 to 65535	0 to 65535	0 to 65535
3	unsigned long	0 to 16777215 16777216 to 4294967295	0 to 16777215 undefined	0 to 16777215	0 to 16777215
4	unsigned long	0 to 4294967295	0 to 4294967295	0 to 4294967295	0 to 4294967295
n>4	unsigned long	0 to 4294967295	0 to 4294967295	0 to 4294967295 > 4294967295	0 to 4294967295 undefined

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Internal Decimal Data Items

The host language variables that are consistent with various sizes of an internal decimal (IDEC) data item are presented in Table 3. This table also shows the largest relative error that can occur when storing data into various sizes of IDEC data items and when retrieving data from various sizes of IDEC data items.

When storing data values into an idec IDEC item, there is no potential for overflow. When retrieving data from an IDEC data item whose size does not exceed eight digits (i.e., $n < 8$), overflow occurs if the stored data value has an absolute value greater than $1.7014118 * 10^{38}$. When retrieving data from an IDEC data item whose size exceeds eight digits (i.e., $n > 8$), overflow occurs if the stored data value has an absolute value greater than $1.7014118 * 10^{38}$.

In instances when the primary host language to the MDBS III data base is C, real numbers (particularly 4 byte and 8 byte real numbers) provide less relative error than IDEC data items. Internal decimal interfaces are present to permit communication (through C) to data bases that are also used with host languages that directly support IDEC variables, such as COBOL.

Table 3. Internal Decimal Correspondences

MDBS IDEC size	C Variable type	Storing Data Largest Relative Error	Retrieving Data Largest Relative Error
1 or 2	float	$5.000 * 10^{-3}$	$2.980 * 10^{-8}$
3 or 4	float	$5.000 * 10^{-5}$	$2.980 * 10^{-8}$
5 or 6	float	$5.000 * 10^{-7}$	$2.980 * 10^{-8}$
7 or 8	float	$5.000 * 10^{-9}$	$2.980 * 10^{-8}$
9 or 10	double	$5.000 * 10^{-11}$	$6.94 * 10^{-18}$
11 or 12	double	$5.000 * 10^{-13}$	$6.94 * 10^{-18}$
n > 12 and n is odd	double	$5.000 * 10^{-(n+2)}$	$6.94 * 10^{-18}$
n > 12 and n is even	double	$5.000 * 10^{-(n+1)}$	$6.94 * 10^{-18}$

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Real Data Items

The host language variables that are consistent with various sizes of a real data item are presented in Table 4. This table also shows the largest relative error that can occur when storing data into various sizes of real data items and when retrieving data from various sizes of real data items.

There is no potential for overflow when storing data into or retrieving data from a real data item. However, there is a possible loss in precision as shown in Table V-4.

Table 4. Real Correspondences

MDBS Real size	C Variable type	Storing Data Largest Relative Error	Retrieving Data Largest Relative Error
2	float	$3.9061906 * 10^{-3}$	0
3	float	$1.5199185 * 10^{-5}$	0
4	float	0	0
5	double	$2.3283063 * 10^{-10}$	0
6	double	$9.0948082 * 10^{-13}$	0
7	double	$3.5388359 * 10^{-15}$	0
8	double	0	0
9	double	0	$1.3823578 * 10^{-17}$
10	double	0	$1.3877576 * 10^{-17}$
11	double	0	$1.3877787 * 10^{-17}$
12	double	0	$1.3877788 * 10^{-17}$
n > 12	double	0	$\frac{256^{-7} - 256}{1-256^{(1-n)}}^{(1-n)}$

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Repeating Data Items

When storing data into or retrieving data from a repeating data item, a C array is used. The appropriate kind of array for each data item type and size is shown below, where m represents the number of replications defined for data item in the DDL specification (with an occurs clause). Here, "data" is the host language array being used for storage or retrieval.

Repeating MDBS Data Item (m replications)		Form of the C Variable	
type	size	Array	data
binary	n	char	data[m][n];
character	n	char	data[m][n];
string	n	char	data[m][n+1];
date	—	char	data[m][10];
time	—	char	data[m][9];
integer	1 or 2	short int	data[m];
integer	≥ 3	long int	data[m];
unsigned	1 or 2	short unsigned	data[m];
unsigned	≥ 3	long unsigned	data[m];
idec	1,2,...,or 8	float	data[m];
idec	≥ 9	double	data[m];
real	2, 3, or 4	float	data[m];
real	≥ 5	double	data[m];

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

DML COMMAND FORM

Data manipulation language commands appear in one of four forms that depend upon the language used with MDBS III. The MDBS III C language interface uses the record/direct DML command form because C supports data structures and permits direct calls to external functions without difficulty. (See the record/direct example for each DML command in the MDBS DMS Manual). The precise command syntax is illustrated in the following examples.

Note: MDBS.DMS expects the data structures found in C and handles all necessary alignments between the structures and data base records during data exhcange.

Command Status

After performing each DML command, the MDBS III data base control system returns a status code to the application program that indicates the success or failure of the command. In UNIX, the command status variable must be an int short.

Special Commands

Two DML special commands can be used in this environment: ALTEOS() and DMSSJP().

ALTEOS

ALTEOS changes the end-of-set command status within an application program from 255 to -1; this allows simpler processing of normal DMS command statuses in your programs. To change the end-of-set command status in a program, you could use the following program fragment:

```
int e0;
      .
      .
      .
 e0 = alteos();
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

DMSSJP

DMSSJP is an automatic error trapping device that monitors the DMS command status returned from each DML request (Chapter XIII of the MDBS III Data Management System Reference Manual describes DMSSJP.). DMSSJP has two arguments, a pointer to an envrionment buffer and a pointer to an array of short integers.

The integer array contains the DMS command status values excluded from error trapping. The last element of the arrray must be 0. When 0 and 255 are the only command status codes excluded from error trapping (normal situations), you can specify NULL as the second DMSSJP argument. To disable all error checking, call DMSSJP with both arguments NULL.

For example, to trap all DMS command status values other than 11, 19, 255, and 0, your application program would include the following program statements:

```
jmp_buf envbuf;
Short exmpcs[4] = {255,11,19,0};
if ((e0 = setjmp(&envbuf)))
{
      .
      .
      .
   error handling code
      .
      .
      .
}
dmssjp(&envbuf,exmpcs);
```

This causes the error handling code to be performed whenever a command status other than 255, 11, 19 or 0 results from a DML request.

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Open and Close Command Examples

The commands to provide access to (open) and finish using (close) a data base from an application program are part of the Utility DML Commands described in chapter XII of the MDBS III Data Management System Reference Manual.

In the example, the elements of the **opblk** structure are consistent with string data items. The element **dbuser** is defined as 17 bytes, to be consistent with a 16 byte string item (discussed under Data Exchange in this appendix). A similar situation holds for the **dbpass**, **dbmode**, and **dbfile** structure elements.

```
int e0;
struct
{
        char dbuser[17];
        char dbpass[13];
        char dbmode[ 5];
        char dbfile[81];
} opblk;
    .
    .
    .
e0 = dbopn(&opblk);
    .
    .
    .
e0 = dbcls();
```

Find Command Examples

The DML find commands locate individual records within the data base. The find commands are fully described in chapter III of the MDBS III Data Management System Reference Manual.

In the example, the variables name and id contain sort key values for the sets **set1** and **set5**. The names **set1** and **set5** identify sets of that have been specified in a DDL specification.

```
int e0;
char name[20];
int id;
    .
    .
    .
e0 = fmsk("set1", &name);
    .
    .
    .
e0 = fmsk("set5", &id);
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Retrieval and Modify Command Examples

The DML retrieval and modify commands provide data transfer capability between the data base and an application program. The retrieval and modify commands are fully described in chapters IV and V of the MDBS III Data Management System Reference Manual.

In the example, the variable **data** is consistent with the data item "ytdearn" in the record that is referenced by the current of run unit.

The data structure **dbrec** is corresponds to the data base record definition of the record defined as the member of set **set2**. The names **ytdearn** and **set2** identify elements of the data base that have been specified in a DDL specification.

The **getc** and **putc** DML commands in this host language have been renamed to **dbgetc()** and **dbputc()** due to name conflicts with C functions.

```
int e0;
float data;
struct {
            char dbfield1[17];
            char dbfield2[26];
            char dbfield3[28];
            char dbfield4[81];
} dbrec;
     .
     .
     .
e0 = gfc("ytdearn", &data);
data = data + 1082.17;
e0 = pfc("ytdearn", &data);
     .
     .
     .
e0 = ofm("set2", dbrec);
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Assignment Command Examples

The DML assignment commands change currency indicator values within the data base and are described in chapter VI of the MDBS III Data Management System Reference Manual.

Here, **set1** and **set3** are names of sets that have been specified in a DDL specification. The variable **i** and the array **blk** identify user defined currency indicators used in the DML assignment commands.

```
int e0;
unsigned short i;
unsigned short blk[2];
      .
      .
      .
blk[0] = 1;
blk[1] = 2;
e0 = suu(blk);
      .
      .
      .
e0 = som("set1,set3");
      .
      .
      .
i=1;
e0 = sou("set1", &i);
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Schema Access Command Examples

The schema access commands, described in chapter XII of the MDBS III Data Management System Reference Manual, provide information about the structure of the data base. The result of these DML commands place portions of the schema information in C structures; the schema access commands do NOT affect currency indicators or alter the data base contents.

In the example, **designrec** receives the design counts for the schema.

```
int e0;
struct design_rec{
     int short       gdc1,
                     gdc2,
                     gdc3,
                     gdc4,
                     gdc5,
                     gdc6,
                     gdc7,
                     gdc8;
     char            gdc9[18],
                     gdc10[18];
} design_info;
     .
     .
e0 = gdc(&design_info);
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Utility Command Examples

The DML utility commands, described in chapter XII of the MDBS III Data Management System Reference Manual, provide several miscellaneous functions to the data base environment that are not addressed by other DML commands.

In the example, **nmem** and **rnam** are variables that receive information about the data base. **nmem** must be an unsigned long variable (that is, consistent with a 4-byte unsigned data item).

The variable **rnam** is 8 bytes since **gtm** requires a variable consistent with an 8 byte character data item. The record type name returned by **gtm** will be in upper case, with blank fill (this same convention also holds for **gto**, **gtc**).

Also, **set1**, **set3**, **SALEMP**, and **possess** are names of data base elements that have been specified in a DDL specification.

```
int e0;
unsigned nmem;
char rnam[8];
     .
     .
e0 = gmc("set1", &nmem);
     .
     .
e0 = gtm("set3", rnam);
     .
     .
e0 = tot("SALEMP, possess");
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Boolean Command Examples

The DML boolean commands are described in chapter XIII of the MDBS III Data Management System Reference Manual and work with groups of data records.

The program example uses the schema diagrammed below and finds the different shipments of a product (widget) to a particular warehouse (Carmel):

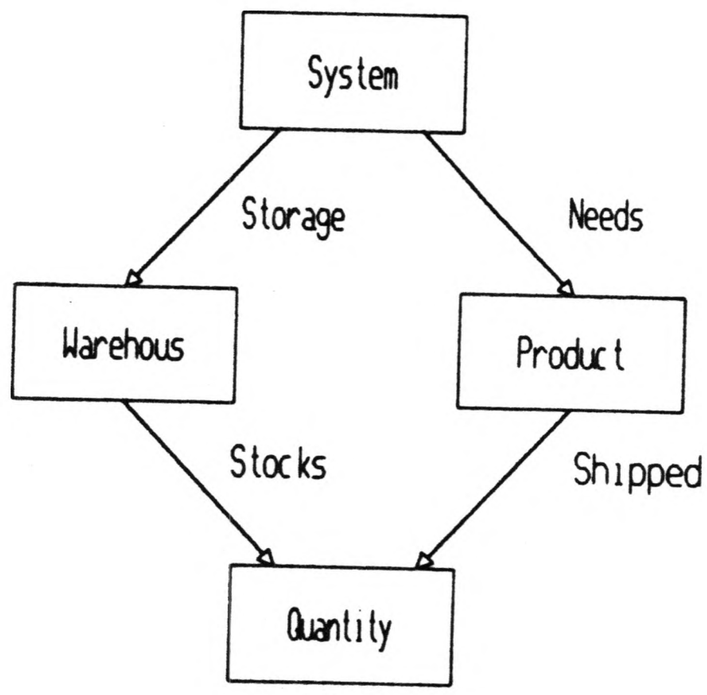

Figure A-1. Boolean Command Example Schema

```
int e0;
unsigned short ind1, ind2;
char whname[16],
     prodname[16];
.
.
.
ind1 = 1;
strncpy("Carmel",&whname,16);
e0 = fmsk("storage",&whname);          /* Locate the Carmel warehouse record */
e0 = som("stocks,storage");
e0 = suo("stocks",&ind1);              /* Set user indicator 1 to the owner of stocks */

strncpy("widget",&prodname,16);
e0 = fmsk("needs",&prodname);          /* Locate the widget record */
e0 = som("shipped,needs");
ind2 = 2;
e0 = suo("shipped",&ind2);             /* Set user indicator 2 to the owner of shipped */
e0 = amm("stocks,shipped,$sysset");    /* Include in $SYSSET each quantity of the */
                                       /* widget shipped to the Carmel warehouse */
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

Syntax of DML Commands

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

The DML command forms for C require all data values transferred between the data base control system and the application to reside in structures. Sample structure declarations and allocations for the DML commands are listed below.

COMMENT - This section declares the C structures used with MDBS III.

```
struct dbrec1{                          /* Data Base Record (A sample) */
                                        /*    Variables corresponding to the DDL record */
                                        /*    definition are declared here. */
        .
        .
        .
};

struct krec1{                           /* Definition for compound sort or calc keys */
     vartype   KFIELD1;                 /*    First field of the key */
     vartype   KFIELD2;                 /*    Next key field */
      .
      .                                 /*    Continue until all key fields are included */
      .
};


struct orec{                            /* Opening Record Definition */
     char  user[16 + 1];
     char pass[12 + 1];
     char mode[4 + 1];
     char filenam[255 + 1];
};

struct area_rec{                        /* Area information (Used for GAI) */
     int short   gai_1;                 /*    Pointer Access */
     int short   gai_2;                 /*    Page Size */
     int short   gai_3;                 /*    Number of Pages */
     char gai_4[8 + 1];                 /*    Area Name */
     char gai_5;                        /*    Filler */
};

struct design_rec{                      /* Design Counts (Used for GDC) */
     int short   gdc_1;                 /*    Number of Defined Areas */
     int short   gdc_2;                 /*    Number of Defined Record Types */
     int short   gdc_3;                 /*    Number of Defined Sets */
     int short   gdc_4;                 /*    Most Data Items in a Record */
     int short   gdc_5;                 /*    Largest Number of Record Types in a Set */
     int short   gdc_6;                 /*    Estimated Size of Largest Record */
     int short   gdc_7;                 /*    Largest Number of Data Items in a Key */
     int short   gdc_8;                 /*    Data Base Key Width (Used for SDC) */
     char gdc_9[18];                    /*    DDL Version Used to Initialize Data Base */
     char gdc_10[18];                   /*    Filler - Presently Unused */
};
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
struct item_rec{
        int short   gii_1;
        int short   gii_2;
        int short   gii_3;
        int short   gii_4;
        int short   gii_5;
        char gii_6[8 + 1];
        char gii_7[1];
};
```

```
/* Data Item Information (Used for GII) */
/*    Size */
/*    Decimal Digits */
/*    Repetitions */
/*    Item Type */
/*    Item Codes */
/*    Item Name */
/*    Filler – Presently Unused */
```

```
struct key_rec{
        int short   gki_1;
        int short   gki_2;
        int short   gki_3;
        int short   gki_4;
        int short   gki_5;
        char gki_6[8 + 1];
        char gki_7;
};
```

```
/* Key Information (Used for GKI) */
/*    Size */
/*    Decimal Digits */
/*    Repetitions */
/*    Item Type */
/*    Item Codes */
/*    Item Name */
/*    Filler – Presently Unused */
```

```
struct rec_rec{
        int short   gri_1;
        int short   gri_2;
        int short   gri_3;
        char gri_4[8 + 1];
        char gri_5[8 + 1];
};
```

```
/* Record Information (Used for GRI) */
/*    Record Codes */
/*    Size */
/*    Data Item Count */
/*    Record Name */
/*    Clustering Set Name */
```

```
struct set_rec{
        int short   gsi_1;
        int short   gsi_2;
        int short   gsi_3;
        int short   gsi_4;
        int short   gsi_5;
        int short   gsi_6;
        char gsi_7[8 + 1];
        char gsi_8[3];
};
```

```
/* Set Information (Used for GSI) */
/*    Type */
/*    Owner Order */
/*    Owner Count */
/*    Member Order */
/*    Member Count */
/*    Set Codes */
/*    Set Name */
/*    Filler – Presently Unused */
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
alteos()                                /* ALTer End Of Set */

amm(cmdstr)                             /* And Members with Members */
     char *cmdstr;                      /* set-1, set-2, set-3 */

amo(cmdstr)                             /* And Members with Owners */
     char *cmdstr;                      /* set-1, set-2, set-3 */

aom(cmdstr)                             /* And Owners with Members */
     char *cmdstr;                      /* set-1, set-2, set-3 */

aoo(cmdstr)                             /* And Owners with Owners */
     char *cmdstr;                      /* set-1, set-2, set-3 */

aui(num)                                /* Allocate User Indicators */
     unsigned short *num;               /* number of user currency indicators to allocate */

ccu(num)                                /* Check Current of run unit against User */
     unsigned short *num;               /* user currency indicator number */

cra(cmdstr, db_rec)                     /* Create Record in Area */
     char *cmdstr;                      /* record, area */
     struct dbrec1 *db_rec;             /* Data base record data */

crs(cmdstr, db_rec)                     /* Create Record and Store */
     char *cmdstr;                      /* record */
     struct dbrec1 *db_rec;             /* Data base record data */

dbcls()                                 /* Data Base CLoSe */

dbclsa(cmdstr)                          /* not available in this environment */

dbcnv(num)                              /* Data Base format CoNVersion */
     unsigned short *num;               /* format option number */

dbenv(num)                              /* Data Base ENVironment */
     unsigned short *num;               /* environment option number */

dbinit()                                /* not available in this environment */

dbldr()                                 /* not available in this environment */

dbgetc(db_rec)                          /* GET data from Current of run unit */
     struct dbrec1 *db_rec;             /* Data base record data */

dbopn(&open_rec)                        /* Data Base OPeN */
          struct orec open_rec;         /* Opening record definition */
```

PRELIMINARY

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
dbopna()                          /* not available in this environment */

dbputc(db_rec)                    /* PUT data to Current of run unit */
     struct dbrec1 *db_rec;       /* Data base record data (db_rec) */

dbsave()                          /* Data Base SAVE */

dbsel()                           /* not available in this environment */

dbstat(stt)                       /* Data Base STATistics */
     unsigned short stt[5];       /* array of statistics items */

dmssjp (envptr,array)             /* DMS Set Jump */
     jmp_buf envptr;              /* environment buffer */
     short array [];              /* exempt command status numbers */

drc()                             /* Delete Record that is Current */

drm(cmdstr)                       /* Delete Record that is Member */
     char *cmdstr;                /* set-name */

dro(cmdstr)                       /* Delete Record that is Owner */
     char *cmdstr;                /* set-name */

fdrk(cmdstr, key_rec)             /* Find Duplicate Record based on Calc Key */
     char *cmdstr;                /* record-name */
     struct krec1 *key_rec;       /* key-list data */

ffm(cmdstr)                       /* Find First Member */
     char *cmdstr;                /* set-name */

ffo(cmdstr)                       /* Find First Owner */
     char *cmdstr;                /* set-name */

ffs(cmdstr)                       /* Find First Sequential record */
     char *cmdstr;                /* area-name (may be null) */

flm(cmdstr)                       /* Find Last Member */
     char *cmdstr;                /* set-name */

flo(cmdstr)                       /* Find Last Owner */
     char *cmdstr;                /* set-name */

fmi(cmdstr, data)                 /* Find Member based on Data Item */
     char *cmdstr;                /* item-name, set-name */
     struct {...} *data;          /* item data */
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
fmsk(cmdstr, key_rec)          /* Find Member based on Sort Key */
     char *cmdstr;             /* set-name */
     struct krec1 *key_rec;    /* sort-key data */

fnm(cmdstr)                    /* Find Next Member */
     char *cmdstr;             /* set-name */

fnmi(cmdstr, data)             /* Find Next Member based on data Item */
     char *cmdstr;             /* item-name, set-name */
     struct {...} *data;       /* item-data */

fnmsk(cmdstr, key_rec)         /* Find Next Memmber based on Sort Key */
     char *cmdstr;             /* set-name */
     struct krec1 *key_rec;    /* sort-key data */

fno(cmdstr)                    /* Find Next Owner */
     char *cmdstr;             /* set-name */

fnoi(cmdstr, data)             /* Find Next Owner based on data Item */
     char *cmdstr;             /* item-name, set-name */
     struct krec1 *data;       /* item-data */

fnosk(cmdstr, key_rec)         /* Find Next Owner based on Sort Key */
     char *cmdstr;             /* set-name */
     struct krec1 *key_rec;    /* sort-key data */

fns(cmdstr)                    /* Find Next Sequential Record */
     char *cmdstr;             /* area-name */

foi(cmdstr, data)              /* Find Owner based on data Item */
     char *cmdstr;             /* item-name, set-name */
     struct {...} *data;       /* item data */

fosk(cmdstr, key_rec)          /* Find Owner based on Sort Key */
     char *cmdstr;             /* set-name */
     struct krec1 *key_rec;    /* sort-key data */

fpm(cmdstr)                    /* Find Prior Member */
     char *cmdstr;             /* set-name */

fpmi(cmdstr, data)             /* Find Prior Member based on data Item */
     char *cmdstr;             /* item-name, set-name */
     struct {...} *data;       /* item data */

fpmsk(cmdstr, key_rec)         /* Find Prior Member based on Sort Key */
     char *cmdstr;             /* set-name */
     struct krec1 *key_rec;    /* sort-key data */
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
fpo(cmdstr)                          /* Find Prior Owner */
     char *cmdstr;                   /* set-name */

fpoi(cmdstr, data)                   /* Find Prior Owner based on data Item */
     char *cmdstr;                   /* item-name, set-name */
     struct {...} *data;             /* item data */

fposk(cmdstr, key_rec)               /* Find Prior Owner based on Sort Key */
     char *cmdstr;                   /* set-name */
     struct krec1 *key_rec;          /* sort-key data */

frk(cmdstr, key_rec)                 /* Find Record based on calc Key */
     char *cmdstr;                   /* record-name */
     struct krec1 *key_rec;          /* key-list data */

gafn(cmdstr, data)                   /* Get Area File Name */
     char *cmdstr;                   /* area-name */
     char *data;                     /* area file name */

gai(cmdstr, area_info)               /* Get Area Information */
     char *cmdstr;                   /* area-name */
     struct area_rec *area_info;     /* area data */

gdc(design_info)                     /* Get Design Counts */
     struct design_rec *design_info;     /* schema design counts */

getc()                               /* use dbgetc() */

getm(cmdstr, db_rec)                 /* GET data from Member */
     char *cmdstr;                   /* set-name */
     struct dbrec1 *db_rec;          /* record-data */

geto(cmdstr, db_rec)                 /* GET data from Owner */
     char *cmdstr;                   /* set-name */
     struct dbrec1 *db_rec;          /* record-data */

gfc(cmdstr, data)                    /* Get Field from Current of run unit */
     char *cmdstr;                   /* item-name */
     struct {...} *data;             /* item-data */

gfm(cmdstr, data)                    /* Get Field from Member */
     char *cmdstr;                   /* item-name, set-name */
     struct {...} *data;             /* item-data */

gfo(cmdstr, data)                    /* Get Field from Owner */
     char *cmdstr;                   /* item-name, set-name */
     struct {...} *data;             /* item-data */
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
ghrv(cmdstr, data)                  /* Get High Range Value */
     char *cmdstr;                  /* record-name, item-name */
     struct *data;                  /* item high range value */

gii(cmdstr, item_info)              /* Get Item Information */
     char *cmdstr;                  /* record-name, item-name */
     struct item_rec *item_info;    /* area file name */

gki(cmdstr, key_info)               /* Get Key Information */
     char *cmdstr;                  /* key specifics */
     struct key_rec *key_info;;     /* area file name */

glrv(cmdstr, data)                  /* Get Low Range Value */
     char *cmdstr;                  /* record-name, item-name */
     struct *data;                  /* item high range value */

gmc(cmdstr, nmem)                   /* Get Member Count */
     char *cmdstr;                  /* set-name */
     int *nmem;                     /* member count */

goc(cmdstr, nown)                   /* Get Owner Count */
     char *cmdstr;                  /* set-name */
     int *nown;                     /* owner count */

gri(cmdstr, rec_info)               /* Get Record Information */
     char *cmdstr;                  /* record specifics */
     struct rec_rec *rec_info;      /* area file name */

gsi(cmdstr, set_info)               /* Get Set Information */
     char *cmdstr;                  /* set specifics */
     struct set_rec *set_info;      /* area file name */

gtc(name)                           /* Get Type of Current of run unit */
     char name[8];                  /* record-name */

gtm(cmdstr, name)                   /* Get Type of Member */
     char *cmdstr;                  /* set-name */
     char name[8];                  /* record-name */

gto(cmdstr, name)                   /* Get Type of Owner */
     char *cmdstr;                  /* set-name */
     char name[8];                  /* record-name */

ims(cmdstr)                         /* Insert Member into Set */
     char *cmdstr;                  /* set-name */

ios(cmdstr)                         /* Insert Owner into Set */
     char *cmdstr;                  /* set-name */
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
lgfile()                          /* not available in this environment */

lgflsh()                          /* LoG file buffer FLuSH */

lgmsg(msg)                        /* LoG file MeSsaGe */
     char *msg;                   /* log message */

mau(num)                          /* Multi-user Active User indicators */
     unsigned short *num;         /* number of user indicators to be locked */

mcc(rtry)                         /* Multi-user Contention Count */
     unsigned short rtry[2];      /* time units and re-tries */

mcf()                             /* Multi-user Current of run unit Free */

mcp()                             /* Multi-user Current of run unit Protect */

mpl()                             /* not available in this enviroment */

mrid(ru_codes)                    /* Multiuser Run-unit IDentifier
     char ru_codes[8];            /* Blocking run-unit information */

mrtf(cmdstr)                      /* Multi-user Record Type Free */
     char *cmdstr;                /* record-name */

mrtp(cmdstr)                      /* Multi-user Record Type Protect */
     char *cmdstr;                /* record-name */

msf(cmdstr)                       /* Multi-user Set Free */
     char *cmdstr;                /* set-name */

msp(cmdstr)                       /* Multi-user Set Protect */
     char *cmdstr;                /* set-name */

nci()                             /* Null all Currency Indicators */

odrk(cmdstr, db_rec)              /* Obtain Duplicate Record based on Calc Key */
     char *cmdstr;                /* record-name */
     struct dbrec1 *db_rec;       /* Data base record data (db_rec) */

ofm(cmdstr,db_rec)                /* Obtain First Member */
     char *cmdstr;                /* set-name */
     struct dbrec1 *db_rec;       /* Data base record data (db_rec) */
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
ofo(cmdstr,db_rec)                  /* Obtain First Owner */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

olm(cmdstr,db_rec)                  /* Obtain Last Member */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

olo(cmdstr,db_rec)                  /* Obtain Last Owner */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

omi(cmdstr, db_rec)                 /* Obtain Member based on Data Item */
     char *cmdstr;                  /* item-name, set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

omsk(cmdstr, db_rec)                /* Obtain Member based on Sort Key */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

onm(cmdstr,db_rec)                  /* Obtain Next Member */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

onmi(cmdstr, db_rec)                /* Obtain Next Member based on data Item */
     char *cmdstr;                  /* item-name, set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

onmsk(cmdstr, db_rec)               /* Obtain Next Memmber based on Sort Key */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

ono(cmdstr,db_rec)                  /* Obtain Next Owner */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

onoi(cmdstr, db_rec)                /* Obtain Next Owner based on data Item */
     char *cmdstr;                  /* item-name, set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

onosk(cmdstr, db_rec)               /* Obtain Next Owner based on Sort Key */
     char *cmdstr;                  /* set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */

ooi(cmdstr, db_rec)                 /* Obtain Owner based on data Item */
     char *cmdstr;                  /* item-name, set-name */
     struct dbrec1 *db_rec;         /* Data base record data (db_rec) */
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
oosk(cmdstr, db_rec)                    /* Obtain Owner based on Sort Key */
     char *cmdstr;                      /* set-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

opm(cmdstr,db_rec)                      /* Obtain Prior Member */
     char *cmdstr;                      /* set-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

opmi(cmdstr, db_rec)                    /* Obtain Prior Member based on data Item */
     char *cmdstr;                      /* item-name, set-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

opmsk(cmdstr, db_rec)                   /* Obtain Prior Member based on Sort Key */
     char *cmdstr;                      /* set-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

opo(cmdstr,db_rec)                      /* Obtain Prior Owner */
     char *cmdstr;                      /* set-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

opoi(cmdstr, db_rec)                    /* Obtain Prior Owner based on data Item */
     char *cmdstr;                      /* item-name, set-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

oposk(cmdstr, db_rec)                   /* Obtain Prior Owner based on Sort Key */
     char *cmdstr;                      /* set-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

ork(cmdstr, db_rec)                     /* Obtain Record based on calc Key */
     char *cmdstr;                      /* record-name */
     struct dbrec1 *db_rec;             /* Data base record data (db_rec) */

pfc(cmdstr, data)                       /* Put Data into Field of Current of run unit */
     char *cmdstr;                      /* item-name */
     struct {...} *data;                /* item-data */

pfm(cmdstr, data)                       /* Put Data into Field of Member */
     char *cmdstr;                      /* item-name, set-name */
     struct {...} *data;                /* item-data */

pfo(cmdstr, data)                       /* Put Data into Field of Owner */
     char *cmdstr;                      /* item-name, set-name */
     struct {...} *data;                /* item-data */

pifd()                                  /* not available in this environment */

putc()                                  /* use dbputc() */
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
putm(cmdstr, db_rec)                    /* PUT data into Member */
     char *cmdstr;                      /* set-name */
     struct dbrec1 *db_rec;             /* record-data */

puto(cmdstr, db_rec)                    /* PUT data into Owner */
     char *cmdstr;                      /* set-name */
     struct dbrec1 *db_rec;             /* record-data */

rms(cmdstr)                             /* Remove Member from Set */
     char *cmdstr;                      /* set-name */

ros(cmdstr)                             /* Remove Owner from Set */
     char *cmdstr;                      /* set-name */

rsm(cmdstr)                             /* Remove All Members */
     char *cmdstr;                      /* set-name */

rso(cmdstr)                             /* Remove All Owners */
     char *cmdstr;                      /* set-name */

scd(key)                                /* Set Current of run unit from Data base key */
     char key[4];                       /* data base key */

scm(cmdstr)                             /* Set Current of run unit based on Member */
     char *cmdstr;                      /* set-name */

scn()                                   /* Set Current of run unit to Null */

sco(cmdstr)                             /* Set Current of run unit based on Owner */
     char *cmdstr;                      /* set-name */

scu(num)                                /* Set Current of run unit based on User indicator */
     unsigned short *num;               /* user currency indicator number */

sdc(key)                                /* Set Data base key to Current of run unit */
     char key[4];                       /* data base key */

setpbf()                                /* not available in this environment */

smc(cmdstr)                             /* Set Member based on Current of run unit */
     char *cmdstr;                      /* set-name */

sme(cmdstr)                             /* Set Member to cru (Exception) */
     char *cmdstr;                      /* set-name */

smm(cmdstr)                             /* Set Member based on Member */
     char *cmdstr;                      /* set-name, set-name */
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
smn(cmdstr)                          /* Set Member to Null */
     char *cmdstr;                   /* set-name */

smo(cmdstr)                          /* Set Member based on Owner */
     char *cmdstr;                   /* set-name, set-name */

smu(cmdstr, num)                     /* Set Owner based on User indicator */
     char *cmdstr;                   /* set-name */
     unsigned short *num;            /* user currency indicator number */

soc(cmdstr)                          /* Set Owner based on Current of run unit */
     char *cmdstr;                   /* set-name */

soe(cmdstr)                          /* Set Owner to cru (Exception) */
     char *cmdstr;                   /* set-name */

som(cmdstr)                          /* Set Owner based on Member */
     char *cmdstr;                   /* set-name, set-name */

son(cmdstr)                          /* Set Owner to Null */
     char *cmdstr;                   /* set-name */

soo(cmdstr)                          /* Set Owner based on Owner */
     char *cmdstr;                   /* set-name, set-name */

sou(cmdstr, num)                     /* Set Owner based on User indicator */
     char *cmdstr;                   /* set-name */
     unsigned short *num;            /* user currency indicator number */

suc(num)                             /* Set User indicator to Current of run unit */
     unsigned short *num;            /* user currency indicator number */

sum(cmdstr, num)                     /* Set User indicator based on Member */
     char *cmdstr;                   /* set-name */
     unsigned short *num;            /* user currency indicator number */

sun(num)                             /* Set User indicator to Null */
     unsigned short *num;            /* user currency indicator number */

suo(cmdstr, num)                     /* Set User indicator based on Owner */
     char *cmdstr;                   /* set-name */
     unsigned short *num;

suu(num)                             /* Set User indicator to User indicator */
     unsigned short num[2];          /* user currency indicator number */

tcn()                                /* Test Current of run unit for Null */
```

© COPYRIGHT 1987 Micro Data Base Systems, Inc.

C Language Appendix

```
tct(cmdstr)
     char *cmdstr;                    /* Test Current of run unit Type */
                                      /* record-name */

tmn(cmdstr)
     char *cmdstr;                    /* Test Member for Null */
                                      /* set-name */

tmt(cmdstr)
     char *cmdstr;                    /* Test Member Type */;
                                      /* record-name, set-name */

ton(cmdstr)
     char *cmdstr;                    /* Test Owner for Null */
                                      /* set-name */

tot(cmdstr)
     char *cmdstr;                    /* Test Owner Type */
                                      /* record-name, set-name */

trabt()                               /* not available in this environment */

trbgn()                               /* TRansaction BeGiN */

trcom()                               /* TRansaction COMmit */

tun(num)
     unsigned short *num;             /* Test User for Null */
                                      /* user currency indicator number */

xmm(cmdstr)
     char *cmdstr;                    /* eXclude Members from Members */
                                      /* set-1, set-2, set-3 */

xmo(cmdstr)
     char *cmdstr;                    /* eXclude Members from Owners */
                                      /* set-1, set-2, set-3 */

xom(cmdstr)
     char *cmdstr;                    /* eXclude Owners from Members */
                                      /* set-1, set-2, set-3 */

xoo(cmdstr)
     char *cmdstr;                    /* eXclude Owners from Owners */
                                      /* set-1, set-2, set-3 */
```


© COPYRIGHT 1987 Micro Data Base Systems, Inc.